

# On the Complexity of Optimization Problems based on Compiled NNF Representations

Daniel Le Berre and Emmanuel Lonca and Pierre Marquis

CRIL-CNRS, Université d'Artois, France  
lastname@cril.fr

## Abstract

Optimization is a key task in a number of applications. When the set of feasible solutions under consideration is of combinatorial nature and described in an implicit way as a set of constraints, optimization is typically NP-hard. Fortunately, in many problems, the set of feasible solutions does not often change and is independent from the user's request. In such cases, compiling the set of constraints describing the set of feasible solutions during an off-line phase makes sense, if this compilation step renders computationally easier the generation of a non-dominated, yet feasible solution matching the user's requirements and preferences (which are only known at the on-line step). In this article, we focus on propositional constraints. The subsets  $L$  of the NNF language analyzed in Darwiche and Marquis' knowledge compilation map are considered. A number of families  $\mathcal{F}$  of representations of objective functions over propositional variables, including linear pseudo-Boolean functions and more sophisticated ones, are considered. For each language  $L$  and each family  $\mathcal{F}$ , the complexity of generating an optimal solution when the constraints are compiled into  $L$  and optimality is to be considered w.r.t. a function from  $\mathcal{F}$  is identified.

## Introduction

Many applications from AI and other domains amount to an optimization task (using e.g. the pseudo-Boolean optimization (Roussel and Manquinho 2009) or MaxSat (Li and Manyà 2009) representation). However, when the set of feasible solutions under consideration is of combinatorial nature, and described in an implicit way as a set of constraints, optimization is typically intractable in the worst case. Fortunately, in many problems, the set of feasible solutions does not often change and is independent from the optimization criterion. As a matter of example, consider the software dependency management problem, and more precisely, the GNU/Linux package dependency management problem (Mancinelli et al. 2006). Constraints are of the form “package  $A$  in version 1 requires package  $B$  in any version” and “package  $A$  in version 2 requires packages  $B$  and  $C$  in any version”, “both versions of package  $A$  cannot be installed together” and can be encoded by propositional formulae like  $\phi = (A \Leftrightarrow (A_1 \vee A_2)) \wedge (B \Leftrightarrow (B_1 \vee B_2)) \wedge (C \Leftrightarrow (C_1 \vee C_2)) \wedge (A_1 \Rightarrow B) \wedge (A_2 \Rightarrow (B \wedge C)) \wedge (\neg A_1 \vee \neg A_2)$ . Given the hard constraint  $\phi$ , an initial state “package  $B$  is installed in version 1”, and more generally some user requirements “install

package  $A$ ”, a dependency solver must find (if possible) a set of packages to be installed such that the constraints and the user requirements are fulfilled. Such a decision problem is NP-complete (Syrjnen 1999; Mancinelli et al. 2006). Considering in addition some user preferences about the packages to be installed leads to an NP-hard optimization problem for which several specific solvers have been designed recently (Tucker et al. 2007; Argelich et al. 2010; Gebser, Kaminski, and Schaub 2011; Janota et al. 2012).

Clearly enough, in the software dependency management problem (as in many configuration problems), all the available pieces of information do not play the same role: all users share the same constraints, as the dependencies between packages do not depend on the user. What makes each user “specific” is the initial state of her system, and more generally her own requirements, as well as the preferences she can have over the feasible solutions. Pursuing the toy example above, the dependency problem admits the two following solutions:  $\{A_1, A, B_1, B\}$ ,  $\{A_2, A, B_1, B, C_1, C\}$ . A user who favors the least changes between the initial and the final states would prefer the first solution to the second one, whereas a user who favors the most recent packages would make the other choice. In such cases, compiling the set of constraints describing the set of feasible solutions during an off-line phase makes sense, if this compilation step renders computationally easier the generation of a non-dominated, yet feasible solution matching the user's requirements and preferences (which are only known at the on-line step).

In the following, we focus on propositional constraints. We consider the languages  $L$  analyzed in (Darwiche and Marquis 2002) as target languages for knowledge compilation. Each of those languages  $L$  satisfies the conditioning transformation, i.e., for each of them, there exists a polynomial-time algorithm which associates with each formula  $\phi \in L$  and each consistent term  $\gamma$  representing a partial assignment a formula from  $L$  representing the conditioning  $\phi \mid \gamma$  of  $\phi$  by  $\gamma$ . This conditioning  $\phi \mid \gamma$  is equivalent to the most general consequence of  $\phi \wedge \gamma$  which is independent of the variables occurring in  $\gamma$ . Equivalently, it is the formula obtained by substituting in  $\phi$  each occurrence of a variable  $x$  of  $\gamma$  by the Boolean constant  $\top$  (resp.  $\perp$ ) if the polarity of  $x$  in  $\gamma$  is positive (resp. negative). The fact that the conditioning transformation is tractable for each of those languages enables to take into account efficiently an

initial state and the user's requirements (on our running example,  $\gamma = A \wedge B_1$ ) during the on-line phase.

We also consider a number of families  $\mathcal{F}$  of representations of objective functions  $f$  over propositional variables, including linear pseudo-Boolean functions, and more sophisticated ones (like polynomial pseudo-Boolean functions (Boros and Hammer 2002)). Here, solutions correspond to propositional interpretations  $\omega$  and criteria are represented by propositional formulae  $\phi_i$ , and are thus of Boolean nature. We note  $\phi_i(\omega) = 1$  when  $\omega$  is a model of  $\phi_i$ , otherwise  $\phi_i(\omega) = 0$ . The importance of a criterion  $\phi_i$  is measured by the weight  $w_i$  associated with it (a real number). Each weight  $w_i$  expresses the penalty of satisfying  $\phi_i$  (it is a cost when positive and a reward otherwise). Each function  $f$  is represented by a weighted base  $\{(\phi_i, w_i) \mid i \in \{1, \dots, n\}\}$ , i.e., a finite multi-set of propositional formulae, where each formula  $\phi_i$  is associated its weight and is also characterized by the aggregator  $\oplus$  used to combine the weights  $w_i$ .  $\omega$  is feasible when it satisfies the hard constraint  $\phi \in L$ ; in such a case,  $f(\omega)$  is defined as  $f(\omega) = \oplus_{i=1}^n w_i \cdot \phi_i(\omega)$ . Two aggregators are considered: a utilitarian one ( $\oplus = \Sigma$ ) and an egalitarian one ( $\oplus = \text{leximax}$ ). An optimal solution  $\omega^*$  is a feasible one which minimizes the value of  $f$ .

The contribution of the paper is a complexity landscape for the optimization problem in such a setting, i.e., the problem of determining an optimal solution (when it exists) given  $\phi \in L$  and a weighted base from  $\mathcal{F}$ . More precisely, for each language  $L$ , each family  $\mathcal{F}$  of representations, and each of the two aggregators, one determines whether this problem can be solved in polynomial time or it cannot be solved in polynomial time unless  $P = NP$ . The problem is considered in the general case, and under the restriction when the cardinality of the weighted base is bounded.

The rest of the paper is organized as follows. After some formal preliminaries, we focus on the family of linear representations of pseudo-Boolean objective functions, characterized by weighted bases for which each  $\phi_i$  is a literal. We show that DNNF and its subsets (Darwiche 1999; Darwiche 2001) are precisely the subsets  $L$  of NNF (among those identified in the knowledge compilation map) for which the optimization problem can be solved in polynomial time. Especially, this problem is NP-hard for all the other subsets in the general case. Afterwards, we switch to the more general family of polynomial representations of pseudo-Boolean objective functions, i.e., when each  $\phi_i$  is a term, and finally to the more general case when each  $\phi_i$  is any NNF formula. For these families, the optimization problem is NP-hard for each language  $L$ , even under some strong restrictions on the representations of the objective function  $f$ , except when  $L = \text{DNF}$ . Finally, we show that some additional tractable cases can be reached (especially for the polynomial representations of  $f$ ) provided that a preset number of criteria is considered (i.e., the cardinality of the weighted base is bounded).

## Formal Preliminaries

In the following, we consider a finite set of propositional variables denoted PS; we sometimes omit it in the notations (when it is not ambiguous to do so).  $\perp$  is the Boolean

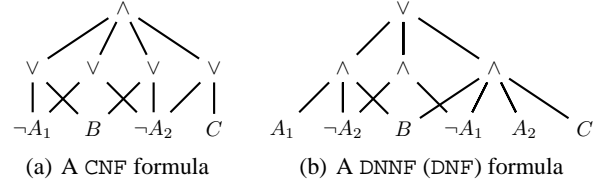


Figure 1: NNF formulae

constant always false, while  $\top$  is the Boolean constant always true.  $l = \neg x$  with  $x \in \text{PS}$  is a negative literal, and  $l = x$  with  $x \in \text{PS}$  is a positive literal. Boolean constants are also considered as positive literals. If  $l = \neg x$  (resp.  $l = x$ ) then its complementary literal  $\sim l$  is  $\sim l = x$  (resp.  $\sim l = \neg x$ ). A complete assignment  $\omega$  of variables in PS is called an interpretation:  $\omega$  is a set of pairs  $(v \in \text{PS}, \{0, 1\})$  where each  $v \in \text{PS}$  is the first projection of exactly one pair in  $\omega$ .  $\omega$  is also viewed as the canonical term  $\bigwedge_{(v,0) \in \omega} \neg v \wedge \bigwedge_{(v,1) \in \omega} v$ .  $\Omega_{\text{PS}}$  is the set of all interpretations over PS. An interpretation  $\omega$  is a model of a formula  $\phi$  if and only if the assignment of the variables of PS in  $\phi$  according to  $\omega$  leads  $\phi$  to be evaluated to 1.  $\phi(\omega)$  denotes the truth value given to  $\phi$  by  $\omega$ . The languages we consider in the following are described in Darwiche and Marquis' compilation map (Darwiche and Marquis 2002). They are subsets of the following NNF language (composed of circuits, alias DAG-shaped "formulae") (Darwiche 1999; Darwiche 2001).

**Definition 1 (NNF)** *NNF is the set of rooted, directed acyclic graphs (DAGs) where each leaf node is labeled with  $\perp$ ,  $\top$ ,  $x$  or  $\neg x$ ,  $x \in \text{PS}$ ; and each internal node is labeled with  $\wedge$  or  $\vee$  and can have arbitrarily many children.*

Figure 1(a) is a NNF formula which represents some parts of the constraints used in our running example. This formula is equivalent to the CNF formula  $(\neg A_1 \vee B) \wedge (\neg A_1 \vee \neg A_2) \wedge (B \vee \neg A_2) \wedge (\neg A_2 \vee C)$ , which is also a NNF formula, since CNF is a subset of NNF. The size of a formula  $\phi$  in NNF, denoted  $|\phi|$ , is the number of arcs in it. For any node  $N$  of a NNF formula  $\phi$ ,  $\text{Vars}(N)$  denotes the set of variables labeling the leaf nodes which can be reached from  $N$ ;  $\text{Vars}(\phi)$  is equal to  $\text{Vars}(N)$  where  $N$  is the root node of  $\phi$ .

In (Darwiche and Marquis 2002), several properties have been considered on the NNF language, leading to a family of languages which are subsets of NNF. We have already noted that CNF is one of these subsets, but many more languages have been considered. In this article, each language  $L$  from the knowledge compilation map, namely NNF, DNNF, d-NNF, s-NNF, f-NNF, d-DNNF, sd-DNNF, BDD, FBDD, OBDD, OBDD<sub><</sub>, DNF, CNF, PI, IP and MODS, is considered. We add to them the languages  $\text{DNNF}_T$  and  $\text{SDNNF}$  which have been introduced later (Pipatsrisawat and Darwiche 2008). For space reasons, we recall here only some of the properties and the corresponding languages. More details are to be found in (Darwiche and Marquis 2002; Pipatsrisawat and Darwiche 2008).

**Definition 2 (decomposability)** *(Darwiche 1999; Darwiche 2001) A NNF formula  $\phi$  satisfies the decomposability property if for each conjunction  $C$  in  $\phi$ , the conjuncts*

of  $C$  do not share variables. That is, if  $C_1, \dots, C_n$  are the children of an and-node  $C$ , then  $\text{Vars}(C_i) \cap \text{Vars}(C_j) = \emptyset$  for  $i \neq j$ .

The subset of NNF which satisfies the decomposability property is called DNNF. Note that the DNF language is a subset of DNNF (assuming without loss of generality that all literals of the terms refer to different variables).

A number of queries and transformations have been considered in the knowledge compilation map. A query corresponds to a computational problem which consists in extracting some information from a given NNF formula  $\phi$ , without modifying it. A transformation aims at generating an NNF formula. Queries and transformations are also viewed as properties which are satisfied or not by a given subset  $L$  of NNF:  $L$  is said to satisfy a given query/transformation precisely when there exists a polynomial-time algorithm for answering the query/achieving the transformation provided that the input is a formula from  $L$ . Among others, the following queries and transformations have been considered in the knowledge compilation map:

- **CO** (consistency): a language  $L$  satisfies the consistency query **CO** if and only if there exists a polynomial-time algorithm that maps every formula  $\phi$  from  $L$  to 1 if  $\phi$  is consistent (i.e.,  $\phi$  has at least one model), and to 0 otherwise;
- **CD** (conditioning): a language  $L$  satisfies the **CD** transformation if and only if there exists a polynomial-time algorithm that maps every formula  $\phi$  from  $L$  and a consistent term  $\gamma$  to a formula of  $L$  that is logically equivalent to  $\phi \mid \gamma$ .

In the following, two aggregation functions are considered: the standard summation  $\Sigma$  aggregator leading to an utilitarian aggregation of values as well as *leximax*, which is a refinement of *max* and leads to an egalitarian aggregation of values (Moulin 1991); when *leximax* is considered, a solution  $\omega$  is considered at least as preferred as a solution  $\omega'$  when  $f(\omega) \leq f(\omega')$ , where  $f(\omega)$  (resp.  $f(\omega')$ ) is the  $n$ -vector of scores associated with  $\omega$  (resp.  $\omega'$ ) and reordered in a decreasing way;  $\leq$  denotes here the lexicographic ordering over the vectors of scores. Thus, one prefers to minimize first the penalties stemming from the most important criteria (the ones of highest weights), then those stemming from the second most important criteria, and so on. For instance, provided that  $\text{PS} = \{A_1, A_2, B_1, C_1\}$ , given the weighted base  $\{(A_2 \wedge C_1, 2), (B_1 \wedge \neg C_1, 1)\}$ ,  $\omega = \{(A_1, 1), (A_2, 0), (B_1, 1), (C_1, 0)\}$  is (strictly) preferred to  $\omega' = \{(A_1, 0), (A_2, 1), (B_1, 1), (C_1, 1)\}$  because  $f(\omega) = (1, 0) < f(\omega') = (2, 0)$ . Unlike  $\Sigma$ , no balance between criteria is possible when *leximax* is used.

In the following we consider that the pseudo-Boolean optimization functions are represented as weighted bases.  $\mathcal{G}$  denotes the set of ("general") representations, i.e., when the  $\phi_i$  are any NNF representations. One also considers several restrictions on the  $\phi_i$  (leading to some subsets of  $\mathcal{G}$ ) which prove of interest:

- *linear representations* are when each  $\phi_i$  is a literal;  $\mathcal{L}$  is the corresponding language.

- *quadratic representations* are when each  $\phi_i$  is a term of size at most 2;  $\mathcal{Q}$  is the corresponding language.
- *polynomial representations* are when each  $\phi_i$  is a term;  $\mathcal{P}$  is the corresponding language.

We have the obvious inclusions:

$$\mathcal{L} \subset \mathcal{Q} \subset \mathcal{P} \subset \mathcal{G}.$$

$\mathcal{Q}$  is of interest for evaluating the complexity of the optimization problem for  $\mathcal{P}$  because Rosenberg proved that every polynomial representation of a pseudo-Boolean function  $f$  can be associated in polynomial time with a quadratic representation of  $f$ , without altering the set of optimal solutions (Rosenberg 1975).

For each language  $\mathcal{F}$  among  $\mathcal{L}, \mathcal{Q}, \mathcal{P}, \mathcal{G}$ , we also consider the subset  $\mathcal{F}^+$  of  $\mathcal{F}$  obtained by assuming that the only literals occurring in any  $\phi_i$  are positive ones. Finally, for each of the resulting languages  $\mathcal{F}$ , we consider the subset  $\mathcal{F}_+$  of it obtained by assuming that each weight  $w_i$  is from  $\mathbb{R}^+$ . Thus, for instance,  $\mathcal{P}^+$  denotes the set of all polynomial representations based on positive literals (i.e., the  $\phi_i$  are positive terms),  $\mathcal{Q}_+$  denotes the set of all quadratic representations with non-negative weights, and  $\mathcal{L}_+^+$  denotes the set of all linear representations, based on positive literals and with non-negative weights.

The optimization query (**OPT**) is defined as follows:

### Definition 3 (OPT)

- **OPT** (optimization): a language  $L$  satisfies **OPT** given  $\mathcal{F}$  and  $\oplus$  if and only if there exists a polynomial-time algorithm that maps every formula  $\phi$  from  $L$  and every representation from  $\mathcal{F}$  of a pseudo-Boolean optimization function  $f$  to an optimal solution of  $\phi$  given  $f$  and  $\oplus$  when a feasible solution exists, and to "no solution" otherwise.

To make things more precise, we note **OPT**[ $L, \mathcal{F}, \oplus$ ] the optimization problem for  $L$  when the weighted base representing  $f$  is in  $\mathcal{F}$  and  $\oplus$  is the aggregator.

It is important to understand that the complexity of the optimization query for  $L$  depends on the *representation* of the pseudo-Boolean optimization function  $f$  (this representation is part of the input), and not on the function  $f$  itself. Indeed, on the one hand, consider any pseudo-Boolean optimization function  $f$  which is represented in an explicit way (i.e., as the weighted base  $\{(\omega, f(\omega)) \mid \omega \in \Omega_{\text{PS}}\}$ ). Any  $L$  in NNF satisfies **OPT** in such a case because  $\Omega_{\text{PS}}$  is part of the input: just consider each  $\omega \in \Omega_{\text{PS}}$  and check in polynomial time whether  $\omega$  is a model of  $\phi$ . If so, compare  $f(\omega)$  with the optimal value *opt* obtained so far and replace *opt* by  $f(\omega)$  if  $f(\omega)$  is better than *opt* (and in this case store  $\omega$ ). On the other hand, consider the case of a constant function  $f$ , represented by an empty weighted base; in this case, solving the optimization problem for  $L$  amounts to determining whether  $L$  satisfies or not **CO**, which is not doable in polynomial time for many subsets of NNF.

In the following, we analyze the computational complexity of **OPT** for the languages  $L$  from the knowledge compilation map satisfying **CO**, namely DNNF, d-DNNF, FBDD, OBDD<sub><</sub>, DNF, IP, PI and MODS, together with DNNF<sub>T</sub>. We ignore OBDD (resp. SDNNF) because only one formula is considered for **OPT**, which prevents from ordering (resp.

vtree) clashes; thus the results will be exactly the same as the ones for  $\text{OBDD}_{<}$  (resp.  $\text{DNNF}_T$ ). As to the representation of weighted bases, one considers the languages  $\mathcal{L}$ ,  $\mathcal{Q}$ ,  $\mathcal{P}$ ,  $\mathcal{G}$ , and their restrictions to positive literals and/or positive weights. Finally, we consider both  $\Sigma$  and *leximax* as aggregators.

The rationale for rejecting languages  $L$  not satisfying **CO** is obvious: if determining whether a feasible solution exists is NP-hard, then determining whether an optimal one exists is NP-hard as well:

**Proposition 1** *If  $L$  does not satisfy **CO** unless  $\text{P} = \text{NP}$ , then  $\text{OPT}[L, \mathcal{F}, \oplus]$  is NP-hard whatever  $\mathcal{F}$  and  $\oplus$ .<sup>1</sup>*

Specifically, this is the case for all languages  $L$  which do not qualify as target languages for knowledge compilation, like CNF and NNF (Darwiche and Marquis 2002).

## Linear Representations

We first consider the case of linear representations of pseudo-Boolean objective functions. A pseudo-Boolean objective function  $f$  is said to be linear when it has a linear representation. Obviously enough, this is not the case for every pseudo-Boolean objective function. Especially, when  $\oplus = \Sigma$  is the aggregator, only modular functions can be represented linearly. Nevertheless, modular functions  $f$  are enough for many problems. Thus, a long list of scenarios where such functions are used is reported in (Boros and Hammer 2002). Clearly,  $\text{OPT}[L, \mathcal{L}, \Sigma]$  is NP-hard for many  $L$ , like in the case of pseudo-Boolean optimization (Roussel and Manquinho 2009) (i.e., when  $L$  consists of conjunctions of equations or inequations over Boolean variables) or more generally in the case of mixed integer programming (Nemhauser and Wolsey 1988).

## Intractable Cases

Let us start with the subsets of NNF for which **OPT** given  $\mathcal{L}$  is intractable. We have already shown that if  $L$  does not satisfy **CO** unless  $\text{P} = \text{NP}$ , then  $\text{OPT}[L, \mathcal{F}, \oplus]$  is NP-hard whatever  $\mathcal{F}$  and  $\oplus$ . Obviously, this hardness result still holds when  $\mathcal{F} = \mathcal{L}$ . The following proposition shows that the converse implication does not hold: it can be the case that  $L$  satisfies **CO**, and that  $\text{OPT}[L, \mathcal{L}, \oplus]$  is NP-hard:

**Proposition 2**  *$\text{OPT}[\text{PI}, \mathcal{L}_+^+, \oplus]$  is NP-hard for  $\oplus = \Sigma$  and  $\oplus = \text{leximax}$ .*

As a consequence,  $\text{OPT}[\text{PI}, \mathcal{F}, \oplus]$  is NP-hard (with  $\oplus = \Sigma$  or  $\oplus = \text{leximax}$ ) for every superset  $\mathcal{F}$  of  $\mathcal{L}_+^+$ .

## Tractable Cases

We now define the concepts of partial interpretation and model generator which will be used in the following lemmas.

**Definition 4 (partial interpretation)** *Let  $V$  be a set of propositional variables ( $V \subseteq \text{PS}$ ). A partial interpretation  $\omega_V$  over  $\text{PS}$  is a set of pairs  $(v \in V, \{0, 1\})$  where each  $v \in V$  is the first projection of exactly one pair in  $V$ . It is also viewed as the term  $\bigwedge_{(v,0) \in \omega_V} \neg v \wedge \bigwedge_{(v,1) \in \omega_V} v$ .*

<sup>1</sup>Proofs of propositions are located at the end of the paper.

The term *partial* comes from the fact that  $\omega_V$  may be extended by adding the missing pairs  $(v \in \text{PS} \setminus V, \{0, 1\})$  to get a “full” interpretation over  $\text{PS}$ . Let  $\omega$  be an interpretation on  $\text{PS}$  and  $\omega_V$  be a partial interpretation on  $\text{PS}$ . If  $\omega_V \subseteq \omega$ , then  $\omega$  is an *extension* of  $\omega_V$ . When all the extensions generated from a partial interpretation  $\omega_V$  satisfy a formula  $\phi$ , and no other interpretation satisfies  $\phi$ ,  $\omega_V$  is said to be a *model generator* of  $\phi$ .

**Definition 5 (model generator)** *Let  $\omega_V$  be a partial interpretation such that the set of its extensions is equal to the set of models of  $\phi$ .  $\omega_V$  is a model generator of  $\phi$ .*

We now describe a polynomial-time algorithm for computing an optimal model of a DNNF formula  $\phi$  given a linear representation of a pseudo-Boolean function. This algorithm returns “no solution” when  $\phi$  is inconsistent. Otherwise, it generates an optimal solution in a bottom-up way, from the leaves to the root of the DNNF formula. The correctness of the optimization algorithm is based on a number of lemmas.

**Lemma 1** *Let  $\omega_V$  be a model generator of an NNF formula  $\phi$ . Given a linear representation of a pseudo-Boolean function  $f$ , one can generate in polynomial time an optimal model  $\omega^*$  of  $\phi$  given  $f$  such that  $\omega^*$  extends  $\omega_V$ .*

Furthermore, given a NNF formula  $\phi$  which reduces to a leaf, computing a model generator of  $\phi$  when it exists and determining that no such generator exists otherwise is easy:

**Lemma 2** *Let  $\phi$  be a NNF formula such that  $|\phi| = 1$ . One can compute in constant time a model generator of  $\phi$  when it exists and determine that no such generator exists otherwise.*

This addresses the base cases. It remains to consider the general case. Let us first focus on decomposable AND nodes. The decomposability property implies that the children of these nodes do not share variables, thus the model generator of a AND node can be derived from the union of the model generators of its children.

**Lemma 3** *Let  $\phi = \bigwedge_{i=1}^k \phi_i$  be a NNF formula rooted at a decomposable AND node. Suppose that each  $\phi_i$  ( $i \in \{1, \dots, k\}$ ) is associated with its model generator  $\omega_i$  when such a generator exists. Then a model generator of  $\phi$  can be computed in linear time from  $\omega_1, \dots, \omega_k$ , when it exists.*

The last kind of DNNF nodes to be considered are OR nodes. OR nodes are the nodes. We highlight here that the model generator of an OR node can be derived from the ones of its children by selecting a model generator leading to the best optimal value. Note that in this case, the model generator does not represent all the models of the formula rooted at the OR node, but a set of models which contain at least one optimal model.

**Lemma 4** *Let  $\phi = \bigvee_{i=1}^k \phi_i$  be a NNF formula rooted at an OR node. Suppose that each  $\phi_i$  ( $i \in \{1, \dots, k\}$ ) is associated with its model generator  $\omega_i$  when such a generator exists. Then, when it exists, for any linear representation of a pseudo-Boolean function  $f$ , one can select in linear time among  $\omega_1, \dots, \omega_k$  a partial interpretation  $\omega_{\text{opt}}$  ( $\text{opt} \in \{1, \dots, k\}$ ) which is extended by an optimal model of  $\phi$  given  $f$ .*

Taking advantage of the previous lemmas, one gets immediately a polynomial-time algorithm for generating an optimal model of a DNNF formula  $\phi$  (when it exists) given a linear representation of a pseudo-Boolean function  $f$ . As a consequence, we get that DNNF satisfies **OPT** given  $\mathcal{L}$ :

**Proposition 3**  *$\text{OPT}[\text{DNNF}, \mathcal{L}, \oplus]$  is in  $\mathbf{P}$  for  $\oplus = \Sigma$  and  $\oplus = \text{leximax}$ .*

Consequently, optimization also is tractable for each subset of DNNF, including DNF, IP, d-DNNF, FBDD, OBDD, MODS and DNNF<sub>T</sub>. In the case  $\oplus = \Sigma$ , this proposition coheres with a result reported in (Kimmig, Van den Broeck, and De Raedt 2012) which shows how to solve in polynomial time the weighted model counting problem when the input is a smooth DNNF formula (that is, for each disjunction node of the DNNF formula, its children mention the same variables). Indeed, it turns out that  $(\mathbb{R}, \min, +, +\infty, 0)$  is a commutative semiring and that, in this semiring, the weighted model count associated with a DNNF formula  $\phi$  given the weights  $\{(l_i, w_i) \mid l_i \text{ literal over PS}\}$  is, under some computationally harmless conditions,<sup>2</sup> equal to the value of an optimal model of  $\phi$  for the pseudo-Boolean function represented by the weighted base  $\{(l_i, w_i) \mid l_i \text{ literal over PS}\}$ . In our case, one not only computes such a value, but also returns an optimal solution  $\omega^*$  leading to this value. Furthermore, our tractability result also applies to the case  $\oplus = \text{leximax}$ . Finally, it is important to note that this tractability result cannot be extended to the full family of  $OWA_W$  aggregators (Yager 1988) (this explains why we focused on specific  $OWA_W$  aggregators, namely  $\Sigma$  and  $\text{leximax}$ ):

**Proposition 4** *Let  $L$  be any subset of NNF containing  $\top$ . For some  $W$ ,  $\text{OPT}[L, \mathcal{L}_+, OWA_W]$  is NP-hard.*

This result is based on a polynomial-time reduction from  $\text{OPT}[L, \mathcal{Q}_+, \Sigma]$  to  $\text{OPT}[L, \mathcal{L}_+, OWA_W]$  for some  $W$  and uses the Proposition 5 found in the next section.

## Non-Linear Representations

Optimization processes are typically considered for making a choice between multiple solutions. In some cases, linear pseudo-Boolean functions are not expressive enough to encode the preference relations of interest. Interestingly, many non-linear optimization functions  $f$  admit linearizations, i.e., one can associate in polynomial time with an optimization problem based on such an  $f$  an optimization problem based on a linear function which has “essentially” the same optimal solutions as the original problem. Such a linearization process is achieved by adding new variables and constraints to the problem; those variables and constraints depend on  $f$ , and “essentially” means here any optimal solution of the linearized problem must be projected

<sup>2</sup> $\phi$  must be consistent, smooth and every variable of PS must occur in it; the consistency condition can be decided in linear time when  $\phi$  is a DNNF formula; furthermore, every DNNF formula can be turned in time linear in it and in the number of variables of PS into a smoothed DNNF formula in which every variable of PS occurs.

onto the original variables to lead to a solution of the initial optimization problem. Such an approach is used in practice in the non-linear track of the pseudo-Boolean evaluation (Roussel and Manquinho 2009). For example, a linearization process can be applied to polynomial representations of pseudo-Boolean optimization functions (see below) by adding a constraint and an integer variable (decomposed using Boolean variables) for each term. A number of optimization procedures have also been proposed for some aggregations of linear representations of pseudo-Boolean functions, for instance when the aggregator under consideration belongs to the  $OWA_W$  family (Ogryczak and Śliwiński 2003; Boland et al. 2006; Galand and Spanjaard 2012); in the corresponding encodings the number of constraints and integer variables to be added are typically quadratic in the number of functions to be aggregated.

Unfortunately, such linearization techniques are not convenient in the general case when the set of feasible solutions is represented by a DNNF formula. Indeed, if one wants to exploit the optimization algorithm presented in the previous section, then it is necessary to compute a DNNF formula equivalent to the conjunction of the DNNF formula coming from the initial optimization problem with the new constraints issued from the linearization process. The point is that such a computation cannot be achieved in polynomial time in the general case (DNNF does not satisfy the conjunctive closure transformation (Darwiche 1999; Darwiche 2001)). The complexity results we present in the following implies that this is actually the case for many non-linear functions, unless  $\mathbf{P} = \mathbf{NP}$ . Accordingly, the benefits offered by the compilation-based approach (i.e., solving efficiently some optimization queries when  $f$  varies) are lost in the general case when non-linear representations are considered.

## Polynomial Representations

Let us start with the polynomial representations of pseudo-Boolean objective functions. Unlike the linear case, every pseudo-Boolean objective function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  has a polynomial representation. Indeed,  $\{(\omega, f(\omega)) \mid \omega \in \Omega_{\text{PS}}\}$  is a polynomial representation of  $f$ .

**OPT** turns out to be NP-hard for almost all valuable subsets of NNF, even under some strong conditions on the representation of the optimization function:

**Proposition 5** *For each subset  $L$  of NNF containing  $\top$  (which is the case for all the subsets of NNF considered in the paper, including MODS),  $\text{OPT}[L, \mathcal{Q}^+, \oplus]$  and  $\text{OPT}[L, \mathcal{Q}_+, \oplus]$  are NP-hard with  $\oplus = \Sigma$  or  $\oplus = \text{leximax}$ , even under the restriction  $\phi = \top$ .*

## General Representations

Clearly enough, as a consequence of the results reported in the previous section, the optimization problem  $\text{OPT}[L, \mathcal{G}, \oplus]$  is intractable in the general case. One way to recover tractability consists in imposing some strong restrictions on both the language  $L$  and the weighted base:

**Proposition 6**  *$\text{OPT}[\text{DNF}, \mathcal{G}_+^+, \oplus]$  is in  $\mathbf{P}$  for  $\oplus = \Sigma$  or  $\oplus = \text{leximax}$ .*

Both restrictions are needed; especially, the result cannot be generalized to the other subsets of DNNF considered in this paper (except of course IP and MODS which are subsets of DNF):

**Proposition 7**  $\text{OPT}[\text{OBDD}_{<}, \mathcal{Q}_+^+, \oplus]$  and  $\text{OPT}[\text{PI}, \mathcal{Q}_+^+, \oplus]$  are NP-hard when  $\oplus = \Sigma$  or  $\oplus = \text{leximax}$ .

### Some Fixed-Parameter Tractability Results

When considering representations of pseudo-Boolean functions based on weighted formulae, a natural restriction is to bound the cardinality  $n$  of the weighted base, since this amounts to considering only a restricted number of criteria of interest. This corresponds to a case for which the user's preferences are, so to say, "simple" ones. It can be expected that the complexity of **OPT** increases as a polynomial in  $n$ . In the following we show that this is the case for the family  $\mathcal{P}$  of polynomial representations (under some harmless conditions on  $L$ ):

**Proposition 8** Let  $L$  be a propositional language that satisfies **CD** and **CO**.  $\text{OPT}[L, \mathcal{P}, \oplus]$  is in FPT with parameter  $n$  for  $\oplus = \Sigma$  and  $\oplus = \text{leximax}$ .

Contrastingly, as soon as we consider general representations of the objective functions, strong assumptions are necessary to get positive results, even when  $n$  is bounded:

**Proposition 9** Provided that each  $\phi_i$  used in the representation of the weighted base is an  $\text{OBDD}_{<}$  representation (resp. a  $\text{DNNF}_T$  representation),  $\text{OPT}[\text{OBDD}_{<}, \mathcal{G}, \oplus]$  (resp.  $\text{OPT}[\text{DNNF}_T, \mathcal{G}, \oplus]$ ) is in FPT with parameter  $n$  for  $\oplus = \Sigma$  and  $\oplus = \text{leximax}$ .

Unfortunately, when considering the representation language  $\mathcal{G}$  in the general case, the results are again mostly negative. While Proposition 6 states that when weighted bases are represented in  $\mathcal{G}_+^+$ , DNF admits a polynomial-time optimization algorithm, it turns out that relaxing any of the two conditions imposed on  $\mathcal{G}$  implies that there is no such algorithm for any of the NNF languages we consider (unless  $\text{P} = \text{NP}$ ), even when  $n$  is bounded:

**Proposition 10** For each subset  $L$  of NNF containing  $\top$ ,  $\text{OPT}[L, \mathcal{G}^+, \oplus]$  and  $\text{OPT}[L, \mathcal{G}_+, \oplus]$  are NP-hard under the restriction  $n \geq 2$  and  $\phi = \top$  for  $\oplus = \Sigma$  and  $\oplus = \text{leximax}$ .

In particular, this proposition shows that  $\text{OPT}[\text{DNF}, \mathcal{G}^+, \oplus]$  and  $\text{OPT}[\text{DNF}, \mathcal{G}_+, \oplus]$  are NP-hard, even when  $n \geq$

2, for  $\oplus = \Sigma$  and  $\oplus = \text{leximax}$ . Hence switching from  $\mathcal{G}_+^+$  to any of its supersets  $\mathcal{G}_+$  or  $\mathcal{G}^+$  has a strong impact on complexity (compare Proposition 10 with Proposition 6).

Finally, we derived the following proposition showing that no result similar to Proposition 6 holds for some other interesting subsets of DNNF which are not subsets of DNF (namely  $\text{OBDD}_{<}$  and  $\text{PI}$ ), even when the cardinality of the weighted base is supposed to be bounded:

**Proposition 11**  $\text{OPT}[\text{OBDD}_{<}, \mathcal{G}_+^+, \oplus]$  and  $\text{OPT}[\text{PI}, \mathcal{G}_+^+, \oplus]$  are NP-hard under the restriction  $n \geq 2$  for  $\oplus = \Sigma$  and  $\oplus = \text{leximax}$ .

### Conclusion

In this article, we investigated the feasibility of a compilation-based approach to optimization, where the set of admissible solutions is represented by a hard constraint (a propositional formula)  $\phi$  which is compiled during an off-line phase, and a set of representations of pseudo-Boolean objective functions  $f$  available only at the on-line phase. Two aggregators have been considered ( $\Sigma$  and  $\text{leximax}$ ). Our main results are summarized in Fig.2.

Our study shows that the optimization query remains intractable in most cases except for linear representations of the objective function. In this case, it makes sense to compile the hard constraint  $\phi$  into a DNNF representation since DNNF is the more succinct language among those considered here, offering tractable optimization. However, we have shown that when forcing the weights and the literals of the objective function to be positive, the optimization problem for DNF becomes tractable. We have also investigated the case when the number of weighted formulae in the representation of the objective function is bounded. We found out that under this hypothesis, the optimization query becomes tractable for the polynomial representations of the objective functions. Finally, it is worth noting that while the languages ADD (Bahar et al. 1993), SLDD (Wilson 2005) and AADD (Sanner and McAllester 2005) of valued decision diagrams can be used for representing and handling pseudo-Boolean objective functions  $f$ , they are not suited to our compilation-based approach to optimization. Indeed, the compilation of  $f$  in any of those languages cannot be achieved in polynomial time in the general case; each time a new objective function  $f$  is considered, a (time-consuming) compilation phase must be undertaken.

	$\mathcal{G}$	$\mathcal{G}^+$	$\mathcal{G}_+$	$\mathcal{G}_+^+$	$\{\mathcal{P}, \mathcal{Q}\}$	$\{\mathcal{P}, \mathcal{Q}\}^+$	$\{\mathcal{P}, \mathcal{Q}\}_+$	$\{\mathcal{P}, \mathcal{Q}\}_+^+$	$\mathcal{L}$	$\mathcal{L}^+$	$\mathcal{L}_+$	$\mathcal{L}_+^+$
DNNF	$\times_{p10}^{p5}$	$\times_{p10}^{p5}$	$\times_{p10}^{p5}$	$\times_{p11}^{p7}$	$\odot_{p8}^{p5}$	$\odot_{p8}^{p5}$	$\odot_{p8}^{p5}$	$\odot_{p8}^{p7}$	$\sqrt{p3}$	$\sqrt{p3}$	$\sqrt{p3}$	$\sqrt{p3}$
(*) $\text{DNNF}_T, \text{OBDD}_{<}$	$\times_{p10}^{p5}$	$\times_{p10}^{p5}$	$\times_{p10}^{p5}$	$\times_{p11}^{p7}$	$\odot_{p8}^{p5}$	$\odot_{p8}^{p5}$	$\odot_{p8}^{p5}$	$\odot_{p8}^{p7}$	$\sqrt{p3}$	$\sqrt{p3}$	$\sqrt{p3}$	$\sqrt{p3}$
DNF, IP, MODS	$\times_{p10}^{p5}$	$\times_{p10}^{p5}$	$\times_{p10}^{p5}$	$\sqrt{p6}$	$\odot_{p8}^{p5}$	$\odot_{p8}^{p5}$	$\odot_{p8}^{p5}$	$\sqrt{p6}$	$\sqrt{p3}$	$\sqrt{p3}$	$\sqrt{p3}$	$\sqrt{p3}$
PI	$\times_{p10}^{p2}$	$\times_{p10}^{p2}$	$\times_{p10}^{p2}$	$\times_{p11}^{p2}$	$\odot_{p8}^{p2}$	$\odot_{p8}^{p2}$	$\odot_{p8}^{p2}$	$\odot_{p8}^{p2}$	$\odot_{p8}^{p2}$	$\odot_{p8}^{p2}$	$\odot_{p8}^{p2}$	$\odot_{p8}^{p2}$

Figure 2: Complexity of **OPT** for subsets of d-NNF, when  $\oplus = \Sigma$  or  $\oplus = \text{leximax}$ .  $\sqrt{\phantom{x}}$  means "satisfies",  $\odot$  means "does not satisfy unless  $\text{P} = \text{NP}$  but is in FPT with parameter  $n$ " and  $\times$  means "does not satisfy unless  $\text{P} = \text{NP}$  and is NP-hard as soon as  $n \geq 2$ ". In each cell, the exponent (resp. subscript) indicates the proposition from which the result reported in the cell comes when  $n$  is unbounded (resp. when  $n$  is bounded). (\*)  $\text{OPT}[\text{OBDD}_{<}, \mathcal{G}, \oplus]$  (resp.  $\text{OPT}[\text{DNNF}_T, \mathcal{G}, \oplus]$ ) is in FPT with parameter  $n$  for  $\oplus = \Sigma$  and  $\oplus = \text{leximax}$  when each  $\phi_i$  is in  $\text{OBDD}_{<}$  (resp.  $\text{DNNF}_T$ ).

## References

- [Argelich et al. 2010] Argelich, J.; Berre, D. L.; Lynce, I.; Silva, J. P. M.; and Rapicault, P. 2010. Solving linux upgradeability problems using boolean optimization. In Lynce, I., and Treinen, R., eds., *LoCoCo*, volume 29 of *EPTCS*, 11–22.
- [Bahar et al. 1993] Bahar, R. I.; Frohm, E. A.; Gaona, C. M.; Hachtel, G. D.; Macii, E.; Pardo, A.; and Somenzi, F. 1993. Algebraic decision diagrams and their applications. In *Proceedings of the International Conference Computer-Aided Design (ICCAD)*, 188–191.
- [Biere et al. 2009] Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2009. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.
- [Boland et al. 2006] Boland, N.; Domínguez-Marín, P.; Nickel, S.; and Puerto, J. 2006. Exact procedures for solving the discrete ordered median problem. *Computers & OR* 33(11):3270–3300.
- [Boros and Hammer 2002] Boros, E., and Hammer, P. L. 2002. Pseudo-boolean optimization. *Discrete applied mathematics* 123(1):155–225.
- [Darwiche and Marquis 2002] Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *J. Artif. Intell. Res. (JAIR)* 17:229–264.
- [Darwiche 1999] Darwiche, A. 1999. Compiling knowledge into decomposable negation normal form. In *IJCAI*, 284–289. Citeseer.
- [Darwiche 2001] Darwiche, A. 2001. Decomposable negation normal form. *Journal of the ACM (JACM)* 48(4):608–647.
- [Galand and Spanjaard 2012] Galand, L., and Spanjaard, O. 2012. Exact algorithms for owa-optimization in multiobjective spanning tree problems. *Computers & OR* 39(7):1540–1554.
- [Gebser, Kaminski, and Schaub 2011] Gebser, M.; Kaminski, R.; and Schaub, T. 2011. aspcud: A linux package configuration tool based on answer set programming. In Drescher, C.; Lynce, I.; and Treinen, R., eds., *LoCoCo*, volume 65 of *EPTCS*, 12–25.
- [Janota et al. 2012] Janota, M.; Lynce, I.; Manquinho, V. M.; and Marques-Silva, J. 2012. Packup: Tools for package upgradability solving. *JSAT* 8(1/2):89–94.
- [Karp 1972] Karp, R. M. 1972. *Reducibility among combinatorial problems*. Springer.
- [Kimmig, Van den Broeck, and De Raedt 2012] Kimmig, A.; Van den Broeck, G.; and De Raedt, L. 2012. Algebraic model counting. Technical Report 1211.4475, ArXiv, Cornell University Library.
- [Li and Manyà 2009] Li, C. M., and Manyà, F. 2009. Maxsat, hard and soft constraints. In Biere et al. (2009). 613–631.
- [2006] Mancinelli, F.; Boender, J.; Di Cosmo, R.; Vouillon, J.; Durak, B.; Leroy, X.; and Treinen, R. 2006. Managing the complexity of large free and open source package-based software distributions. In *Automated Software Engineering, 2006. ASE’06. 21st IEEE/ACM International Conference on*, 199–208. IEEE.
- [1991] Moulin, H. 1991. *Axioms of cooperative decision making*. Cambridge University Press.
- [1988] Nemhauser, G. L., and Wolsey, L. A. 1988. *Integer and combinatorial optimization*. Wiley interscience series in discrete mathematics and optimization. Wiley.
- [2003] Ogryczak, W., and Śliwiński, T. 2003. On solving linear programs with the ordered weighted averaging objective. *European Journal of Operational Research* 148(1):80–91.
- [2008] Pipatsrisawat, K., and Darwiche, A. 2008. New compilation languages based on structured decomposability. In *Proc. of AAAI’08*, 517–522.
- [1975] Rosenberg, I. 1975. Reduction of bivalent maximization to the quadratic case. *Cahiers du Centre d’études de Recherche Operationnelle* 17:71–74.
- [2009] Roussel, O., and Manquinho, V. M. 2009. Pseudo-boolean and cardinality constraints. In Biere et al. (2009). 695–733.
- [2005] Sanner, S., and McAllester, D. A. 2005. Affine algebraic decision diagrams (AADDs) and their application to structured probabilistic inference. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 1384–1390.
- [1999] Syrjinen, T. 1999. A rule-based formal model for software configuration. Master’s thesis, Helsinki University of Technology.
- [2007] Tucker, C.; Shuffelton, D.; Jhala, R.; and Lerner, S. 2007. Opium: Optimal package install/uninstall manager. In *ICSE*, 178–188. IEEE Computer Society.
- [2005] Wilson, N. 2005. Decision diagrams for the computation of semiring valuations. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 331–336.
- [1988] Yager, R. R. 1988. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man, and Cybernetics* 18(1):183–190.

## Appendix: Proof Sketches

**Proof 1 (Proposition 1)** Associate in polynomial time with any formula  $\phi$  from  $L$  the hard constraint  $\phi$  and the empty weighted base.  $\phi$  is consistent if and only if the output of the optimization algorithm is not "no solution".

**Proof 2 (Proposition 2)** Every positive Krom formula (i.e., a CNF formula  $\phi$  consisting of binary clauses, where every variable occurs in  $\phi$  only positively) can be turned into an equivalent  $\text{PI}$  formula in polynomial time where all literals are positive. The NP-hardness of optimization in this case comes from the NP-hardness of the minimal hitting set problem, where the cardinality of the sets is 2. Let  $C$  be a set of sets  $c_i$  (where  $|c_i| = 2$ ) whose elements are included in a reference set  $E$ . A hitting set of  $C$  is a set  $h$  of elements of  $E$  such that  $E \cap c_i \neq \emptyset$ . Determining whether a hitting set  $h$  of  $C$  containing at most  $k$  elements exists or not has been proved NP-complete (Karp 1972). Clearly, such a hitting set  $h$  exists if and only if any optimal solution  $\omega^*$  of the optimization problem given by the hard constraint  $\phi = \bigwedge_{c_i \in C} \bigvee_{x \in c_i} x$  (a positive Krom formula) and the weighted base  $\{(x, 1) \mid x \in PS\}$  (which is in  $\mathcal{L}_+^+$ ) is such that  $f(\omega^*) \leq q$  when  $\oplus = \Sigma$ , and  $f(\omega^*)$  contains at most  $k$  ones when  $\oplus = \text{leximax}$ .

**Proof 3 (Lemma 1)** Given a model generator  $\omega_V$  of  $\phi$  and a linear weighted base  $\{(l_i, w_i) \mid i \in \{1, \dots, n\}\}$ , it is easy to compute an optimal model of  $\phi$  which extends  $\omega_V$ : consider successively every literal  $l_i$  the variable  $v_i$  of which is not assigned by  $\omega_V$ ; if  $w_i < 0$ , then add  $(v_i, 0)$  to  $\omega_V$  if  $l_i$  is a negative literal and  $(v_i, 1)$  to  $\omega_V$  otherwise; if  $w_i \geq 0$ , then add  $(v_i, 0)$  to  $\omega_V$  if  $l_i$  is a positive literal and  $(v_i, 1)$  to  $\omega_V$  otherwise.

**Proof 4 (Lemma 2)** There are four kinds of NNF formulae of size 1: if  $\phi = \top$ , then the models of  $\phi$  are the extensions of the model generator  $\{\}$ ; if  $\phi = \perp$ , then  $\phi$  has no model, so no model generator of  $\phi$  exists. If  $\phi = x \in PS$ , then the models of  $\phi$  are the extensions of the model generator  $\{(x, 1)\}$ ; finally, if  $\phi = \neg x$  with  $x \in PS$ , then the models of  $\phi$  are the extensions of the model generator  $\{(x, 0)\}$ .

**Proof 5 (Lemma 3)** If one of the  $\phi_i$  is inconsistent, then it has no model generator, and as a consequence,  $\phi$  has no model generator. In the remaining case, thanks to the decomposability property, the union of the model generators of all the children  $\phi_i$  is a model generator of  $\phi$ .

**Proof 6 (Lemma 4)** The inconsistency of  $\phi$  can be easily detected since  $\phi$  is inconsistent if and only if all the children  $\phi_i$  of  $\phi$  are inconsistent, hence do not have model generators. In the remaining case, since  $\phi$  is a disjunction, the models of  $\phi$  are the union of the models of its children. Hence, every optimal model of  $\phi$  given  $f$  is an optimal model of at least one of its children. Altogether, this implies that at least one of the children  $\phi_i$  of  $\phi$  is associated with a model generator which can be extended to an optimal model of  $\phi$  given  $f$ . Thanks to Lemma 1, one computes in polynomial time an optimal model  $\omega_i^*$  of each  $\phi_i$  from the associated generator; it is then enough to compare the values  $f(\omega_i^*)$  to determine an  $\omega_i^*$  which is an optimal model of  $\phi$ , and to select the corresponding model generator.

**Proof 7 (Proposition 3)** Direct by structural induction on  $\phi$  given Lemmas 1, 2, 3 and 4.

**Proof 8 (Proposition 4)** We point out a polynomial-time reduction from  $\text{OPT}[L, \mathcal{Q}_+, \Sigma]$  to  $\text{OPT}[L, \mathcal{L}_+, \text{OWA}_W]$  for some  $W$ . The existence of such a reduction is enough to get the result given Proposition 5. Let us recall that given a  $s$ -vector of real numbers  $W = (p_1, \dots, p_s)$  such that  $\sum_{i=1}^s p_i = 1$ ,  $\text{OWA}_W$  is the mapping associating with any  $s$ -vector of real numbers  $V = (v_1, \dots, v_s)$  the real number given by  $\text{OWA}_W(V) = \sum_{i=1}^s p_i v_{\sigma(i)}$  where  $\sigma$  is the permutation of  $\{1, \dots, n\}$  such that  $v_{\sigma(1)} \geq \dots \geq v_{\sigma(n)}$ . Consider now any  $\mathcal{Q}_+$  representation of an objective function  $f$  of the form of the weighted base  $\{(l_{i,1} \wedge l_{i,2}, w_i) \mid i \in \{1, \dots, n\}\}$ . Let  $K = \max(\{w_i, i \in \{1, \dots, n\}\}) + 1$ . We associate in linear time with this weighted base the following weighted base  $\{(l_{i,1}, n[K(n-i+1) + w_i]), (l_{i,2}, n[K(n-i+1) + w_i]), (\sim l_{i,1}, nK(n-i+1)), (\sim l_{i,2}, nK(n-i+1)) \mid i \in \{1, \dots, n\}\}$ , and we consider the  $\text{OWA}_W$  aggregator given by the  $4n$ -vector of real numbers  $W = (0, \frac{1}{n}, 0, \frac{1}{n}, \dots, 0, \frac{1}{n}, 0, 0, \dots, 0)$  starting with the sequence  $0, \frac{1}{n}$  repeated  $n$  times and followed by a sequence of  $2n$  zeroes. Let  $g$  be the objective function represented by such a base when this aggregator is considered. We want to prove that for any pair of interpretations  $\omega, \omega'$ , we have  $f(\omega) \leq f(\omega')$  if and only if  $g(\omega) \leq g(\omega')$ . Consider any interpretation  $\omega$  and for any  $i \in \{1, \dots, n\}$  the four weighted formulae  $(l_{i,1}, n[K(n-i+1) + w_i]), (l_{i,2}, n[K(n-i+1) + w_i]), (\sim l_{i,1}, nK(n-i+1)), (\sim l_{i,2}, nK(n-i+1))$ . By construction for each  $j \in \{1, 2\}$ ,  $\omega$  satisfies  $l_{i,j}$  precisely when it does not satisfy  $\sim l_{i,j}$ . Observe also that  $nK(n-i+1)$  (and a fortiori  $n[K(n-i+1) + w_i]$ ) is strictly greater than  $n[K(n-(i+1)+1) + w_{i+1}]$  for each  $i \in \{1, \dots, n-1\}$  since  $K$  is strictly greater than each  $w_{i+1}$ . Accordingly, the vector of values sorted in non-increasing way induced by  $\omega$  and the base representing  $g$  has the form  $(a_1, b_1, \dots, a_n, b_n, 0, \dots, 0)$  where for each  $i \in \{1, \dots, n\}$ ,  $a_i = b_i = n[K(n-i+1) + w_i]$  when  $\omega$  satisfies  $l_{i,1} \wedge l_{i,2}$ ,  $a_i = n[K(n-i+1) + w_i]$  and  $b_i = nK(n-i+1)$  when  $\omega$  satisfies  $\sim l_{i,1} \wedge l_{i,2}$  or  $\omega$  satisfies  $l_{i,1} \wedge \sim l_{i,2}$ , and  $a_i = b_i = nK(n-i+1)$  when  $\omega$  satisfies  $\sim l_{i,1} \wedge \sim l_{i,2}$ . Thus, for each  $i \in \{1, \dots, n\}$ ,  $b_i = n[K(n-i+1) + w_i]$  when  $\omega$  satisfies  $l_{i,1} \wedge l_{i,2}$  (otherwise  $b_i = nK(n-i+1)$ ). Since the vector  $W$  starts with the sequence  $0, \frac{1}{n}$  repeated  $n$  times, the value of  $g(\omega)$  is independent from the  $a_i$  and depends only of the  $b_i$ . More precisely, we have  $g(\omega) = (\sum_{i=1}^n w_i \cdot (l_{i,1} \wedge l_{i,2})(\omega)) + (\sum_{i=1}^n K(n-i+1)) = f(\omega) + K \frac{n(n+1)}{2}$ . Since  $K \frac{n(n+1)}{2}$  is a constant term independent from  $\omega$ , we get the expected result.

**Proof 9 (Proposition 5)**

• **OPT** $[L, \mathcal{Q}_+, \oplus]$ . Consider the following decision problem. Given a finite set  $S = \{\gamma_1, \dots, \gamma_n\}$  of terms of size 2 and an integer  $k$ , one wants to determine whether there exists an interpretation satisfying at least  $k$  terms of  $S$ . This problem is known as NP-complete. Now, we can associate in polynomial time with any  $S$  and  $k$  the instance of **OPT** $[L, \mathcal{Q}_+, \oplus]$  given by  $\phi = \top$ , and the weighted base  $\{(\sim l_{i,1} \wedge l_{i,2}, 1), (l_{i,1} \wedge \sim l_{i,2}, 1), (\sim$



$l_{i,1} \wedge \sim l_{i,2}, 1) \mid \gamma_i = l_{i,1} \wedge l_{i,2} \in S\}$ . By construction, the optimal solution  $\omega^*$  of the instance of  $\mathbf{OPT}[L, \mathcal{Q}_+, \oplus]$  is such that  $f(\omega^*) \leq n - k$  when  $\oplus = \Sigma$  and  $f(\omega^*)$  contains at least  $k$  zeroes (hence at most  $n - k$  ones) when  $\oplus = \text{leximax}$  if and only if there exists an interpretation (namely,  $\omega^*$ ) satisfying at least  $k$  terms of  $S$ . This proves that  $\mathbf{OPT}[L, \mathcal{Q}_+, \oplus]$  is NP-hard for  $\oplus = \Sigma$  and  $\oplus = \text{leximax}$ .

- **$\mathbf{OPT}[L, \mathcal{Q}_+, \oplus]$ .** Consider the the minimal hitting set problem, where the cardinality of the sets is 2, as in the proof of Proposition 2. With each element  $x$  of the reference set  $E = \bigcup_{c_i \in C} c_i$  of cardinality  $n$  corresponds a propositional variable  $x$  meaning that  $x$  is not selected in the hitting set. We can associate in polynomial time with  $C$ , the given collection of  $m$  subsets  $c_i$  of  $E$  and the constant  $k$ , the instance of  $\mathbf{OPT}[L, \mathcal{Q}_+, \oplus]$  given by  $\phi = \top$  and the weighted base  $\{(x_{i,1} \wedge x_{i,2}, 2), (x_{i,1}, -1), (x_{i,2}, -1) \mid c_i = \{x_{i,1}, x_{i,2}\} \in C\}$  in which every occurrence of a multi-occurent pair is removed but one (so that the resulting multi-set actually is a set containing  $m+n$  weighted formulae). Observe that for each  $c_i = \{x_{i,1}, x_{i,2}\} \in C$  and every interpretation  $\omega$ , the "contribution" of the restriction over  $\{x_{i,1}, x_{i,2}\}$  of  $\omega$  to  $f(\omega)$  conveyed by the weighted formulae  $(x_{i,1} \wedge x_{i,2}, 2), (x_{i,1}, -1), (x_{i,2}, -1)$  associated with  $c_i$  consists of the aggregation of values 2, -1 and 0: when both variables  $x_{i,1}$  and  $x_{i,2}$  are set to 1 (i.e., no element of  $c_i$  is selected so that the resulting set will not be a hitting set) the obtained values are 2, -1, -1, when one of them only is set to 1, the obtained values are 0, 0, -1, and finally when both variables are set to 0, the obtained values are 0, 0, 0. By construction the interpretation assigning every variable to 0 always is a hitting set of  $C$  (this hitting set is equal to  $E$ ). When an interpretation corresponds to a hitting set of  $C$ , the values to be aggregated are only among 0 and -1. The optimal solution  $\omega^*$  of the instance of  $\mathbf{OPT}[L, \mathcal{Q}_+, \text{leximax}]$  is such that  $f(\omega^*)$  contains as many -1 as possible (and no 2). Thus,  $C$  has a hitting set of size at most  $k$  if and only if  $n + f(\omega^*) \leq k$  (when  $\oplus = \Sigma$ ) and  $f(\omega^*)$  contains at least  $n - k - 1$  (when  $\oplus = \text{leximax}$ ).

**Proof 10 (Proposition 6)** Let us consider a general representation of the objective function, with positive literals and positive weights  $\{(\phi_i, w_i) \mid i \in \{1, \dots, n\}\}$ . When each  $w_i$  is positive, since a minimal solution is targeted and given the aggregators which are considered, an interpretation which does not satisfy  $\phi_i$  (hence leading to a weight 0) is always preferred to an interpretation satisfying  $\phi_i$  and leading to a weight  $w_i \geq 0$ . Furthermore, positive formulae  $\phi_i$  are monotone: if an interpretation  $\omega$  is a model of  $\phi_i$  then every interpretation which coincides with  $\omega$  on the variables set to 1 by  $\omega$  also is a model of  $\phi_i$ .

Let  $\phi = \bigvee_{j=1}^m t_j$  be a DNF formula. A polynomial-time algorithm for  $\mathbf{OPT}[\text{DNF}, \mathcal{Q}_+, \oplus]$  is as follows. If  $\phi$  contains no consistent term, then return "no solution". Otherwise, for each consistent term  $t_j$  of  $\phi$ , let us define  $\omega_{t_j}$  as the interpretation which satisfies  $t_j$  and sets to 0 every variable which does not occur in  $t_j$ . By construction, among the interpretations  $\omega$  satisfying  $t_j$ , if  $\omega$  does not satisfy  $\phi_i$ , then

necessarily this is also the case of  $\omega_{t_j}$ . Thus  $\omega_{t_j}$  is an optimal solution among the models of  $t_j$ . Since the models of  $\phi$  are exactly the interpretations satisfying at least one  $t_j$ , it is enough to compare in a pairwise fashion the values  $f(\omega_{t_j})$  for each  $t_j$  of  $\phi$  to determine an  $\omega_{t_j}$  which is optimal for  $\phi$ , and finally to return it.

**Proof 11 (Proposition 7)** We have proved (Proposition 5) that  $\mathbf{OPT}[L, \mathcal{Q}_+, \oplus]$  is NP-hard for  $\oplus = \Sigma$  and for  $\oplus = \text{leximax}$ , even when the hard constraint  $\phi = \top$  belongs to  $L$ , which is the case when  $L = \text{OBDD}_{<}$  or  $L = \text{PI}$ . In such a case, we point out a polynomial-time reduction from  $\mathbf{OPT}[\text{OBDD}_{<}, \mathcal{Q}_+, \oplus]$  to  $\mathbf{OPT}[\text{OBDD}_{<}, \mathcal{Q}_+^+, \oplus]$  and from  $\mathbf{OPT}[\text{PI}, \mathcal{Q}_+, \oplus]$  to  $\mathbf{OPT}[\text{PI}, \mathcal{Q}_+^+, \oplus]$  problem. It consists in associating in linear time with any  $\mathcal{Q}_+$  representation a  $\mathcal{Q}_+^+$  representation obtained by replacing every negative literal  $\neg x$  occurring in it by a new variable  $n_x$  ( $X$  denotes the set of variables occurring in such literals). Then one replaces  $\phi = \top$  by an  $\text{OBDD}_{<}$  (resp. a  $\text{PI}$ ) representation of  $\bigwedge_{x \in X} (\neg x \Leftrightarrow n_x)$ . The point is that an  $\text{OBDD}_{<}$  (resp. a  $\text{PI}$ ) representation of  $\bigwedge_{x \in X} (\neg x \Leftrightarrow n_x)$  can be computed in time linear in the input size. Finally, by construction, both optimization problems have the same optimal solutions (once projected on the initial variables).

**Proof 12 (Proposition 8)** Let  $\phi \in L$  and  $\{(t_i, w_i) \mid i = \{1, \dots, n\}\}$  the  $\mathcal{P}$  representation of the objective function. There are  $2^n$  sets of the form  $\{t_i^* \mid i = \{1, \dots, n\}\}$  where each  $t_i^*$  is either  $t_i$  or the clause equivalent to  $\neg t_i$  and obtained as the disjunction of the negations of the literals occurring in  $t_i$ . With each of those sets corresponds a  $n$ -vector of reals  $(v_1, \dots, v_n)$  where each  $v_i$  ( $i \in \{1, \dots, n\}$ ) is equal to  $w_i$  when  $t_i$  is in the set, and to 0 otherwise. When  $n$  is bounded, the number of those sets is bounded as well. Furthermore, when interpreted conjunctively, each set can be viewed as a CNF formula  $\alpha$  which contains at most  $n$  clauses of size  $> 1$ . Each of the clauses in  $\alpha$  contains at most  $m$  literals (where  $m$  is the cardinality of  $\mathcal{P}$ ). As a consequence, each  $\alpha$  can be turned in time  $O(m^n)$  into an equivalent DNF formula  $\alpha'$ , which contains at most  $O(m^n)$  consistent terms. When  $L$  satisfies **CD** and **CO**, we can easily check whether  $\alpha' \wedge \phi$  is consistent by determining whether there exists a consistent term  $t$  in  $\alpha'$  so that  $\phi \mid t$  is consistent. In the enumeration process, every  $\alpha'$  such that  $\alpha' \wedge \phi$  is inconsistent is simply skipped. For each remaining  $\alpha'$  one can easily compute the value  $f(\omega_{\alpha'})$  where  $\omega$  is any model of  $\alpha'$ , hence any model of  $\alpha$ . Indeed, by construction, it is equal to the aggregation function  $\oplus$  applied to the  $n$ -vector of reals  $(v_1, \dots, v_n)$  associated with  $\alpha$ . Hence at each step of the enumeration it is enough to memorize one of the  $\alpha'$  encountered so far leading to a minimal value  $f(\omega_{\alpha'})$ . Once all the  $\alpha$  have been considered, if no  $\alpha$  such that  $\alpha' \wedge \phi$  is consistent has been found, the algorithm returns "no solution"; in the remaining case, an "optimal"  $\alpha'$  has been stored and every interpretation extending a consistent term  $t$  in  $\alpha'$  so that  $\phi \mid t$  is consistent is an optimal solution.

**Proof 13 (Proposition 9)** One takes advantage of a linearization process here. Basically the approach consists in replacing each  $\phi_i$  which is not a literal by a new variable  $n_{\phi_i}$  and in replacing the hard constraint  $\phi$  by an  $\text{OBDD}_{<}$

representation equivalent to  $\phi \wedge \bigwedge_{i=1}^n (n_{\phi_i} \Leftrightarrow \phi_i)$ . The key point is that this OBDD<sub><</sub> representation can be computed in  $\mathcal{O}(|\phi| \cdot (2 \max_{i=1}^n |\phi_i| + 1)^{n-1})$  time because the conjunction of a bounded number of OBDD<sub><</sub> representations can be computed in polynomial time as an OBDD<sub><</sub> representation and an OBDD<sub><</sub> representation of  $n_{\phi_i} \Leftrightarrow \phi_i$  can be computed in time polynomial in the size of the OBDD<sub><</sub> formula  $\phi_i$ , whatever  $<$ . The proof for DNNF<sub>T</sub> is similar (the point is that DNNF<sub>T</sub> satisfies the same bounded conjunction transformation as OBDD<sub><</sub>).

**Proof 14 (Proposition 10)** We consider the problem of determining the consistency of a CNF formula of the form  $\psi^+ \wedge \psi^-$  where every clause of  $\psi^+$  is positive (i.e., it consists of positive literals only) while every clause of  $\psi^-$  is negative (i.e., it consists of negative literals only). It is well-known that this restriction of CNF-SAT is NP-complete.

- **OPT**[ $L, \mathcal{G}^+, \oplus$ ]. We can associate in polynomial time with every CNF formula of the form  $\psi^+ \wedge \psi^-$  the instance of **OPT**[ $L, \mathcal{G}^+, \oplus$ ] given by  $\phi = \top$  and the weighted base  $\{(\psi^+, -1)(\neg\psi^-, 1)\}$  (note that both  $\psi^+$  and  $\neg\psi^-$  belongs to  $\mathcal{G}^+$ ). The point is that  $\psi^+ \wedge \psi^-$  is consistent if and only if the optimal solution  $\omega^*$  of **OPT**[ $L, \mathcal{G}^+, \oplus$ ] which is computed satisfies  $f(\omega^*) = -1$  when  $\oplus = \Sigma$  and  $f(\omega^*) = (0, -1)$  when  $\oplus = \text{leximax}$ .
- **OPT**[ $L, \mathcal{G}_+, \oplus$ ]. We can associate in polynomial time with every CNF formula of the form  $\psi^+ \wedge \psi^-$  the instance of **OPT**[ $L, \mathcal{G}_+, \oplus$ ] given by  $\phi = \top$  and the weighted base  $\{(\neg\psi^+, 1)(\neg\psi^-, 1)\}$ . The point is that  $\psi^+ \wedge \psi^-$  is consistent if and only if the optimal solution  $\omega^*$  of **OPT**[ $L, \mathcal{G}_+, \oplus$ ] which is computed satisfies  $f(\omega^*) = 0$  when  $\oplus = \Sigma$  and  $f(\omega^*) = (0, 0)$  when  $\oplus = \text{leximax}$ .

**Proof 15 (Proposition 11)** The result is a consequence of the fact that **OPT**[ $L, \mathcal{G}_+, \oplus$ ] is NP-hard for  $\oplus = \Sigma$  and  $\oplus = \text{leximax}$  even when the hard constraint  $\phi$  is  $\top$  and  $n \geq 2$  (cf. Proposition 10). Similarly to the reduction used in the proof of Proposition 7, one associates in linear time with any  $\mathcal{G}_+$  representation a  $\mathcal{G}_+^+$  representation obtained by replacing every negative literal  $\neg x$  occurring in it by a new variable  $n_x$  ( $X$  denotes the set of variables occurring in such literals). Then one considers as the new hard constraint an OBDD<sub><</sub> (resp. a PI) representation  $\phi$  of  $\bigwedge_{x \in X} (\neg x \Leftrightarrow n_x)$ . The point is that an OBDD<sub><</sub> (resp. a PI) representation of  $\bigwedge_{x \in X} (\neg x \Leftrightarrow n_x)$  can be computed in time linear in the input size. By construction, both optimization problems have the same optimal solutions (once projected on the initial variables).